Remarks

Upon entry of the foregoing amendment, claims 1-3, 5-13, 22, 26-33, 35-37, and 45-56 are pending in the application, with claims 1, 22, 31, and 45 being the independent claims.

Based on the following remarks, Applicants respectfully request that the Examiner reconsider all outstanding objections and rejections and that they be withdrawn.

Rejections Under 35 U.S.C. § 103

The current Office Action states on page 4 that claims 1-3, 5-13, 22, 26-33, 35-37, and 45-56 are rejected under U.S.C. § 103 as allegedly being unpatentable over U.S. Pat. No. 5,862,066 to Rossin *et al.* (hereinafter, "Rossin") in view of Marc Olano, "A Programmable Pipeline for Graphics Hardware," PhD Dissertation, Department of Computer Science, University of North Carolina, Chapel Hill, April 1998 (hereinafter, "Olano"), and U.S. Pat. No. 5,874,969 to Storm, *et al.* (hereinafter, "Storm"). Applicants respectfully traverse this rejection. Even if these references are combined for the sake of argument, such a combination still does not teach or suggest the claimed invention. Neither Rossin, Olano, nor Storm, taken alone or in combination, teaches or suggests a floating point frame buffer as recited in the claimed invention.

Summary of the Claimed Invention

The present application discloses and claims a computer system and method for graphics processing that can operate in the floating point format throughout the various stages in the graphics processing pipeline. The floating point format can be used to perform geometric calculations and rasterization, as well as to store data in a frame buffer. The data

can also be scanned out (or read) from the frame buffer in the floating point format for further graphics calculations or for display. As discussed in the Background section of the specification, although floating point graphics calculations were performed in previous systems (see page 3, lines 2-6, of the specification), floating point frame buffering and scan out was not done for various reasons. (See pages 4-8 of the specification.)

Summary of the Rossin Reference

Rossin describes a computer graphics system that includes a geometry accelerator, a rasterizer and a frame buffer. (See Rossin, col. 2, lines 13-15.) The Examiner concedes on page 5 of the Office Action that "Rossin fails to explicitly teach 'a floating point frame buffer.'"

Summary of the Olano Reference

The Olano dissertation describes an abstract pipeline on which to model interactive graphics machines for user-written procedures, such as procedural shading. The dissertation includes an introduction, a description of an abstract pipeline, and a description of the PixelFlow graphics hardware system. The document also describes surface shading and lighting stages on the PixelFlow system and describes an implementation of primitive and interpolation stages. The document also discusses user experience of those who have written shading procedures for use on the PixelFlow system and concludes by discussing areas of future research.

Olano's abstract pipeline is introduced in Chapter 2 and shown in Figure 2.1. According to Figure 2.1, the abstract pipeline includes the following stages: model,

transform, primitive, interpolate, shade (and light), atmospheric, and image warp. According to Olano, "each stage can be implemented as a procedure." However, Olano also states that hardware systems may not be able to support every programmed stage. (See Olano, p. 16.) In Chapter 3, Olano maps the abstract pipeline to the PixelFlow graphics hardware system. The mapped PixelFlow system includes a host workstation, various rendering nodes (handling the modeling, transformation, primitive, and interpolation stages), various shading nodes (handling the shading, lighting, and atmospheric stages), and a frame buffer node (handling the image warping stage). This is shown in Figures 3.1, 3.2, and 3.3. There is no detailed discussion of the frame buffer node or usage of a frame buffer. There is also no detailed discussion involving the scanning or reading out data from the frame buffer for display. The remainder of the document appears to focus only on the shading/lighting and primitive/interpolation stages.

Chapter 4 of the Olano document discusses surface shading and the *pfman* shading language, which is said to be similar to the RenderMan shading language. Chapter 4 discusses data types used for shading, and specifically states that the floating point format is supported by the RenderMan, *pfman*, and Mendelbrot shading languages. However, Olano states that "[o]ur pixel processors do not support floating point in hardware, so every floating point operation is built from basic integer math operations" (see Olano, p. 69). Thus, Olano's shader's usage of the floating point format appears to be emulated in software. At the end of Chapter 4 (Olano, pp. 79-80), various rendering, shading, and frame buffering PixelFlow stages are shown in Figure 4.20. The frame buffer stage is described as handling "copying the incoming image pixels into the frame buffer, including any required warping." There is no discussion of data type in this description.

Chapter 5 appears to discuss the primitive and interpolation stages. In section 5.2.1 on page 86, two other graphics pipelines are introduced. Generally, they include the stages of transforming, clipping, rasterizing, interpolating, and shading/texturing (see Figure 5.2). Subsequent to shading/texturing is "display," which does not appear to be considered a programmable stage. There is no discussion of data type in this description.

In the conclusions and discussion of future research of Chapter 7, image warping is discussed in section 7.1.6 on page 118. That discussion states that "PixelFlow has a testbed interface for new procedural code to run on the frame buffer board. However, there is only a single procedural hook to handle both image warping and the operations to load pixels into the frame buffer itself. This latter code is non-trivial and make new versions of this stage difficult to write. For usable support of procedural image warping, this stage should be split into a procedural warping stage and a separate frame buffer stage, hidden from the user." This section does not discuss data types or detailed usage of the frame buffer.

Summary of the Storm Reference

Storm describes a graphics accelerator that includes a command block, a plurality of floating point processors, and one or more draw processors. Storm does not disclose a floating point frame buffer.

Storm describes floating point blocks that "receive high level drawing commands and generate graphics primitives" and "perform transformations, clipping, lighting, and set-up operations on received geometry data" (See Storm, col. 5, lines 43-49). The floating point blocks connect to drawing blocks that perform screen space rendering of the various graphics

primitives. The draw blocks render an image into a frame buffer according to a draw packet received from one of the floating point blocks (Storm, col. 6, lines 19-32).

Differences Between Claimed Invention and Teachings of Cited References

The Examiner concedes on page 6 of the Office Action that "Rossin fails to explicitly teach 'a floating point frame buffer,'" but argues that Storm teaches a floating point frame buffer. Storm provides no such teaching. While Storm's floating point blocks presumably process geometry data and generate primitives using floating point values, Storm never discusses the format of the data that is output by the draw blocks, and never discusses the format of the data stored in the frame buffer. Storm does not state or suggest that the frame buffer stores image values in a floating point format.

The Examiner also argues on page 6 that one of ordinary skill in the art would be motivated to combine Rossin and Olano, thereby modifying the frame buffer of Rossin to achieve floating point precision. On the contrary, there is no such motivation, inasmuch as Olano repeatedly advocates against the use of floating point processing, in favor of fixed point processing. Olano discusses the wide range in possible values that can be represented by fixed point, but states that "For some quantities used in shading this range is overkill. For colors, an 8 to 16 bit fixed point representation is sufficient [Hill97]. But floating point takes four bytes, regardless of range. Worse, there are cases where floating point has too much range but not enough precision." (See Olano, p. 59.) He goes on to state that "In addition to their memory advantages, we can achieve significant speed improvements by using fixed point operations instead of floating point." (Olano, p. 69)

Discussion of Examiner Remarks

The Examiner has rejected independent claims 1, 22, 31, and 45. Each of these claims recites a frame buffer that stores floating point data, or the writing of floating point data to a frame buffer. The Examiner has rejected these claims under 35 U.S.C. 103, arguing that a frame buffer that stores floating point data is disclosed by Storm, and suggesting that such a frame buffer is also disclosed by Olano.

While Storm discloses a graphics accelerator that processes floating point values, there is no disclosure of a frame buffer that stores floating point values. Nor does Olano make such a disclosure.

The Examiner further argues that, with respect to claims 1, 22, and 45, Storm teaches a display screen coupled to a frame buffer in the floating point format. Storm does not teach a frame buffer that stores floating point values, and therefore does not teach a display that reads from such a frame buffer.

Moreover, the Examiner relies on the combination of the Storm, Rossin, and Olano references in arguing that these claims are obvious. Olano, however, repeatedly teaches away from the use of floating point values in graphics processing, as discussed above. Hence a person of skill in the art would not combine Olano with the other references, given Olano's statements as to the shortcomings of floating point representations.

Claims 1, 22, 31, and 45 are therefore not rendered obvious by a combination of the cited references. The feature of a frame buffer that receives and stores floating point values occurs in each of these claims, but is not disclosed by Storm, contrary to the assertions of the Examiner. Moreover, a person of skill in the art would not combine Olano with either Storm or Rossin in order to derive a system or method featuring a frame buffer that receives and

stores floating point values, given that Olano teaches away from the use of floating point values in graphics processing. For at least these reasons, claims 1, 22, 31, and 45 are not obvious over a combination of Rossin, Storm, and Olano.

Each of the dependent claims (i.e., claims 2, 3, 5-13, 32, 33, 35-37, and 46-56) depends from one of the independent claims 1, 22, 31, and 45. Each dependent claim therefore includes the feature of a frame buffer that receives and stores floating point values. Because this feature is not disclosed or suggested by the cited art, none of the dependent claims is rendered obvious over any reasonable combination of the cited art. Nor would a person of skill in the art combine the Olano reference with either of the Storm or Rossin references. For at least these reasons, none of dependent claims 2, 3, 5-13, 32, 33, 35-37, and 46-56 are rendered obvious over the cited art.

Conclusion

All of the stated grounds of objection and rejection have been properly traversed, accommodated, or rendered moot. Applicants therefore respectfully request that the Examiner reconsider all presently outstanding objections and rejections and that they be withdrawn. Applicants believe that a full and complete reply has been made to the outstanding Office Action and, as such, the present application is in condition for allowance. If the Examiner believes, for any reason, that personal communication will expedite prosecution of this application, the Examiner is invited to telephone the undersigned at the number provided.

Prompt and favorable consideration of this Amendment and Reply is respectfully requested.

Respectfully submitted,

STERNE, KESSLER, GOLDSTEIN & FOX P.L.L.C.

Edward W. Yee
Attorney for Applicants
Registration No. 47,294

Date: MAR. 15, 2006

1100 New York Avenue, N.W.
Washington, D.C. 20005-3934
(202) 371-2600